**Low-Level Design**

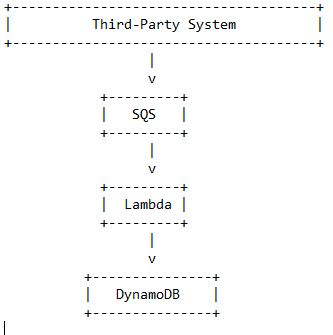
**Project 3**

**Capture transaction data from SQS queue**

**Infrastructure Components:**

* AWS SQS: Asynchronous message queue service for receiving live transaction data from a third-party system.
* AWS Lambda: Serverless compute service for processing and handling incoming messages.
* DynamoDB: NoSQL database for storing Account Master and Source System Master data.

**Project Flow Diagram:**

****

**Detailed Workflow:**

**a. Third-Party System Integration:**

* The live transaction data is posted by the third-party system to the SQS queue in JSON format.

**b. Message Processing:**

* AWS Lambda functions are triggered by messages from the SQS queue.
* The Lambda function retrieves and dequeues the message from the queue.
* The JSON message is parsed and validated to extract the relevant transaction details.

**c. Transaction Creation:**

* The Lambda function interacts with DynamoDB to fetch the necessary data from the Account Master and Source System Master tables.
* Based on the transaction details, three separate transactions are created for each incoming transaction.
* The transactions are recorded in the Ledger Transaction table in DynamoDB.

**d. Transaction Credit:**

* The created transactions are credited to the respective accounts based on the transaction type and amount.
* The "Product Sale" account, "Value Added Tax" account, and "Excise Duty" account are credited accordingly.

**e. Error Handling:**.

* Failed transactions are logged and recorded in a separate file for future reloading purposes.

**Data Models:**

**a. Account Master Table:**

* Table Name: Acct\_master
* Columns:
* acc\_no: Text (Unique, Alphanumeric) - Represents the account number.
* acc\_name: Text - Provides the display name for the account.
* acc\_desc: Text - Stores the description of the accounts.
* acc\_type: Text - Indicates the account type (Income, Expense, Asset, Liability).

**b. Ledger Transaction Table:**

* Table Name: ledger\_txn
* Columns:
* txn\_id: Integer (Sequential, Auto-generated) - Serves as the unique identifier for each transaction.
* voucher\_code: Char(10) (Unique) - Captures the voucher code associated with the transaction.
* txn\_type: Char(1) - Denotes the transaction type (Debit or Credit).
* txn\_date: Date (YYYY-MM-DD) - Represents the date of the transaction.
* acc\_no: Foreign Key (References: Acct\_master.acc\_no) - Links the transaction to the corresponding account.
* txn\_amt: Numeric(11,2) - Stores the transaction amount.
* source\_system\_id: Integer - Identifies the system from which the transaction was posted (Inventory Management System or Self-generated Transaction).
* source\_system\_txn\_id: Varchar(50) - Stores the transaction ID from the source system.

**c. Source System Master Table:**

* Table Name: Src\_sys\_mst
* Columns:
* system\_id: Integer (Sequential, Auto-generated) - Serves as the unique identifier for each source system.
* system\_name: Varchar(30) - Provides a description of the various data source systems.

**Error Handling:**

* Failed transactions are logged and recorded in a separate file for future reloading purposes.
* The failed records file can be stored in a designated location, such as an S3 bucket, for easy access and analysis.

**Maintenance and Reporting:**

* Regular maintenance tasks should be performed to ensure the system's availability, performance, and data integrity.
* Reporting mechanisms can be implemented to generate transaction reports, monitor system performance, and track any errors or failures.

The provided Low-Level Design (LLD) offers a detailed representation of the system's components, workflow, data models, error handling, and maintenance/reporting aspects. However, it is essential to adapt the design based on specific requirements, architectural guidelines, and organizational standards to achieve an optimal implementation.